

2023012321 王锦浩

1. 操作系统: 是控制和管理计算机系统内各种硬件和软件资源, 有效地组织多道程序运行的系统软件, 是用户与计算机之间的接口。
2. 实时操作系统: 是指系统能及时响应外部事件的请求, 在规定的时间内完成对该事件的处理, 并控制所有实时任务协调一致地运行的操作系统。
3. 互斥共享: 指系统中的某些资源, 在一段时间内只能被一个进程访问, 其他进程若要访问该资源, 必须等待, 直到占用该资源的进程释放该资源后才能访问。

1. 硬件系统; 软件系统 3. 硬件资源; 软件资源 5. 共享性; 异步性

第二章

3. 临界区: 每个进程中访问临界资源的那段代码称为临界区。
4. 进程同步: 多个相关进程在运行次序上的协调, 使并发执行的诸进程之间能按照一定的规则共享资源和相互合作, 从而使程序的执行具有可再现性。

1. 动态性, 独立性, 异步性

5. 动态: 静态 6. 间接制约 7. 共享存储器: 消息传递: 管道通信
9. 就绪: 运行: 阻塞。

①

```

2. semaphore S2 = 0, S3 = 0, S4 = 0, S5 = 0, S6 = 0, S7 = 0;
   process S1() {
       // S1 操作
       V(S2);
       V(S3);
   }
   process S2() {
       P(S2);
       // S2 操作
       V(S4);
   }
   process S3() {
       P(S3);
       // S3 操作
   }
   process S4() {
       P(S4);
       // S4 操作
       V(S7);
   }
   process S5() {
       P(S5);
       // S5 操作
       V(S6);
   }
   process S6() {
       P(S6);
       // S6 操作
       V(S7);
   }
   process S7() {
       P(S7);
       // S7 操作
   }

3. semaphore mutex = 1, empty = 5, apple = 0, orange = 0;
   process father() {
       while (1) {
           // 父子水果 (苹果或橙子)
           P(empty);
           P(mutex);
           if (放入苹果) {
               V(apple);
           } else {
               V(orange);
           }
           V(mutex);
       }
   }
   process son() {
       while (1) {
           P(orange);
           P(mutex);
           V(empty);
           V(mutex);
       }
   }
   process daughter() {
       while (1) {
           P(apple);
           P(mutex);
           V(empty);
           V(mutex);
       }
   }

```

(2)

2023012331 王冠宇

4. semaphore empty=1, full=0, empty2=1, full2=0;

```
process PA() {
    while (1) {
        P(empty1);
        V(full1);
    }
}
```

```
process PB() {
    while (1) {
        P(full1);
        P(empty2);
        V(full2);
        V(empty1);
    }
}
```

```
process PC() {
    while (1) {
        P(full2);
        V(empty2);
    }
}
```

t.semaphore mutex=1, empty=N, customers=0, barber=0;

```
payment=0;
process barber() {
    while (1) {
        P(customers);
        P(mutex);
        V(barber);
        V(mutex);
        P(payment);
    }
}
```

```
process customer() {
    P(mutex);
    if (empty > 0) {
        empty--;
        V(customers);
        V(mutex);
        P(barber);
        P(mutex);
        empty++;
        V(payment);
        V(mutex);
    } else {
        V(mutex);
    }
}
```

3

第3章:

2. 处理机调度: 是指根据一定的算法从就绪队列中选择一进程, 将时间片分配给它, 使其执行的过程。
3. 周转时间: 是指从作业提交给操作系统开始, 到作业完成为止的这段时间间隔。
4. 死锁: 指在一组进程中的各个进程均占有不会释放的资源, 但又都请求被其它进程所占而不会释放的资源, 从而导致各进程都处于无休止的等待状态。

2. 提交: 设备: 完成

6. 死锁: 避免死锁: 检测死锁: 解除死锁

5. 算法	进程	完成时间	周转时间	带权周转时间
FCFS	A	3	3	$\frac{1}{3}$
	B	9	7	$\frac{2}{3}$
	C	13	9	$\frac{3}{4}$
	D	18	12	$\frac{3}{2}$
	E	20	12	6

平均周转时间: 7.6

平均带权周时: 3.314

非抢占 SJF	进程	完成时间	周转时间	带权周转时间
	A	3	3	$\frac{1}{3}$
	B	9	7	$\frac{2}{3}$
	C	11	3	$\frac{1}{2}$
	D	15	11	$\frac{1}{4}$
	E	20	14	$\frac{14}{5}$

平均周时: 7.6

平均带权周时: 2.77

10ms - 15ms = 25ms

故按T正

算法	进程	完成时间	周转时间	带权周转时间
短作业优先	A	3	3	$\frac{1}{6}$
	B	9	7	$\frac{1}{4}$
	C	10	7	$\frac{1}{2}$
	D	14	14	$\frac{14}{5}$

平均周转时间 7
平均带权周转时间 2.513

算法	进程	完成时间	周转时间	带权周转时间
HRRN	A	3	3	$\frac{1}{6}$
	B	13	9	$\frac{1}{4}$
	C	15	7	$\frac{1}{2}$
	D	20	14	$\frac{14}{5}$

平均周转时间 8
平均带权周转时间 2.897

算法	进程	完成时间	周转时间	带权周转时间
RR (时间片为1)	A	3	3	1
	B	11	9	$\frac{3}{2}$
	C	14	10	$\frac{5}{2}$
	D	19	13	$\frac{13}{5}$
	E	10	2	1

平均周转时间 7.4
平均带权周转时间 2.43

2. 任务的松弛时间 = 20ms - 0ms - 10ms = 10ms.

B = 50ms - 0ms - 10ms = 40ms.

C = 50ms - 0ms - 15ms = 35ms.

故优先执行A.

10ms $\div 3$

B = 50ms - 10ms - 10ms = 30ms

C = 50ms - 10ms - 15ms = 25ms.

故接下来执行C.

(E)

25ms后C开始执行。

$$A = 20ms - 20ms - 10ms = -10ms$$

$$B = 50ms - 35ms - 10ms = 5ms$$

此时B有剩余正数时间故B开始执行。

综上所述执行的顺序是：任务A → 任务C → 任务B。

3(1) 根据公式: $Allocation = MAX - Need$ 计算得

进程	Allocation		
P0	0	0	32
P1	1	0	00
P2	1	3	54
P3	0	3	32
P4	0	0	14

利用安全性算法对上面状态分析有: 找到3-个安全序列P0, P2, P4, P1, P3
故系统是安全的

12) P2发出请求向量 Request(1, 2, 2, 2)后, 按银行家算法进行检查:

$$\textcircled{1} Request(1, 2, 2, 2) \leq Need(2, 3, 5, 6)$$

$$\textcircled{2} Request(1, 2, 2, 2) \leq Available(1, 6, 2, 2)$$

③ 系统先假定可为P2分配资源, 并修改 Available, Allocation 和 Need 向量
 $Available = (0, 4, 0, 0)$ $Allocation_2 = (2, 5, 7, 6)$ $Need_2 = (1, 1, 3, 4)$

④ 进行安全性检查: $Need_2 \leq Available(0, 4, 0, 0)$ 不成立,

即 Available 不能满足任何进程请求, 故系统进入不安全状态,

因此 P2 发出 Request(1, 2, 2, 2) 后, 不能将资源分配给他

13) 系统立即满足 P2 的请求(1, 2, 2, 2)后, 并没有马上进入死锁状态, 因为此时上述进程没有申请新资源, 并因得不到资源进入阻塞状态, 只有当上述进程提出新的请求, 导致所有没在运行的多个进程因得不到资源而阻塞并形成循环等待时, 系统才进入死锁状态。

⑥